DECISION SCIENCES INSTITUTE

A Comparison of Dispatching Heuristics in a Permutation Flow Shop to Minimize Total Earliness and Tardiness with Unforced Idle Time Allowed

Jeffrey Schaller
Eastern Connecticut State University
schallerj@ecsu.ctstateu.edu

Jorge M. S. Valente
Universidade do Porto
jvalente@fep.up.pt

ABSTRACT

This paper considers the problem of scheduling jobs in a permutation flow shop with the objective of minimizing total earliness and tardiness. Unforced idle time is considered in order to reduce the earliness of jobs. It is shown how unforced idle time can be inserted on the final machine. Several dispatching heuristics that have been used for the problem without unforced idle time were modified and tested. The Modified Due Date procedure (MDD) consistently performs best. An improvement procedure is also proposed and tested and shown to deliver improved solutions.

KEYWORDS: Scheduling, Heuristics, Flow Shop, Earliness and Tardiness

INTRODUCTION

Organizations place a heavy emphasis on just-in-time delivery today. Inventory is the result of early delivery and ties up cash as well as space and resources needed to maintain and manage inventory. Late delivery can cause lost sales or loss of customer good will. This paper addresses this trend by considering an objective that sums the penalties for earliness and tardiness in a flow shop environment. Most research on flow shops has assumed the sequence of jobs to process will be the same on each machine. These schedules are referred to as permutation schedules. This is done for two reasons. First it simplifies the computational effort and second it is often not practical to change the sequence of jobs from one machine to the next. In this research only permutation schedules are considered.

Formally, suppose there is a set of n jobs to be processed in a flow shop with M machines. Let d_j be the due date of job j ($j = 1, \ldots, n$). Let p_{jm} and C_{jm} represent the processing time and completion time of job j ($j = 1, \ldots, n$) on machine m ($m = 1, \ldots, M$). The earliness of job j, E_j, is defined as: $E_j = \max\{d_j - C_{jM}, 0\}$, for $j = 1,\ldots, n$ and the tardiness of job j, T_j, is defined as: $T_j = \max\{C_{jM} - d_j, 0\}$, for $j = 1,\ldots, n$. The objective function, Z, can be expressed as: $Z = \sum_{j=1}^{n} E_j + T_j$.

Since the objective in the problem is non-regular inserting unforced idle time into a schedule for the jobs can help to reduce the earliness of some jobs and thus improve the objective. However, to our knowledge, previous research has not considered unforced idle time for this problem. In this research schedules with unforced inserted idle time are considered. Therefore, if the job to be sequenced in position j is denoted as $[j]$ and $C_{[0]1} = 0$ then $C_{[j]1} = C_{[j-1]1} + I_{[j]1} + p_{[j]1}$ and $C_{[j]m} = \max\{C_{[j]m-1}, C_{[j-1]m}\} + I_{[j]m} + p_{[j]m}$ for $m = 2, \ldots, M$, where $I_{[j]m}$ is the unforced idle time inserted before the job in position [j] on machine m.

LITERATURE REVIEW

Literature Review of Earliness and Tardiness

The earliness and tardiness objective has been studied extensively. Reviews include (Baker and Scudder, 1990) and (Hoogeveen, 2005). The single machine environment is the most commonly considered. We consider the use of inserted idle time and a review (Kanet and Sridharan, 2000) covers scheduling models with inserted idle time.

Literature Review of Earliness and Tardiness in Flow Shops

Research for the earliness and tardiness objective in flow shops has not been extensive and none of the papers considers unforced idle time. Papers that consider some variant of the earliness and tardiness objective in flow shops without unforced idle time include: (Moslehi et al., 2009; Chandra et al., 2009; Madhushini et al., 2009; Zegordi et al., 1995; Rajendran, 1999; Schaller and Valente, 2013b; M'Hallah, 2014; Schaller and Valente, 2013a, and Valente and Schaller, 2017). Valente and Schaller, 2017 tested dispatching heuristics for the problem when unforced idle is not allowed.

INSERTING UNFORCED IDLE TIME TO REDUCE EARLINESS

In order to specify a solution for this problem a sequence for the processing of jobs, as well as, inserting unforced idle is required.

It can be shown that only inserting unforced idle on the final machine need be considered. There are single-machine timetabling procedures for inserting idle time into a given sequence for minimizing total earliness and tardiness that can be modified to accomplish this (Fry et al., 1987, Davis and Kanet, 1993, Kim and Yano, 1994) with the constraint that that a job cannot start on the final machine until it is completed on the previous machine.

DISPATCHING HEURISTICS TESTED

(Valente and Schaller, 2017) tested six dispatching heuristics for minimizing total earliness and tardiness in permutation flow shops but without unforced idle time are considered in this research. These six procedures were modified to consider unforced idle time and are tested in this research.

The following notation is used to describe the procedures. Let S be the current partial schedule and $C_j(S)$ be the completion time of job $j \notin S$ if j is scheduled at the end of S. Let $s_j(S)$ be the slack of job $j \notin S$ if j is scheduled at the end of S, where $s_j(S) = d_j - C_j(S)$. Additionally, let t_m (S) be the current availability time of machine m under schedule S. For convenience, the current time on the first machine will also be denoted by t, so $t = t_1(S)$. Finally, let $P_j(S) = C_j(S) - t$

be the total time (total processing time plus any eventual forced idle time) between the start and finish of job $j \notin S$ if j is scheduled at the end of S.

Earliest Due Date (EDD)

The earliest due date (EDD) was first proposed by (Jackson, 1955). This rule schedules the jobs in non-decreasing order of their due dates d_j.

Modified Earliest Due Date (MDD)

In the modified due date (MDD) heuristic (Baker and Bertrand, 1982; Vepsalainen and Morton, 1987), at each iteration we select the job with the minimum value of the modified due date.

$$MDD_j(S) = max\{d_j, C_j(S)\} = max\{d_j, t + P_j(S)\} = max\{d_j - t, P_j(S)\}.$$

Minimum Slack (SLK) and Minimum Slack Per Work Rule (SLKP)

The minimum slack (SLK) rule (Panwalkar & Iskander, 1977; Vepsalainen & Morton, 1987) chooses, at each iteration, the job with the minimum slack $s_j(S) = d_j - C_j(S)$. The minimum slack per required time (SLK/P) (Panwalkar & Iskander, 1977; Vepsalainen & Morton, 1987) selects, at each iteration, the job with the minimum value of the ratio $SLK/P_j(S) = s_j(S)/P_j(S)$.

LIN-ET Rules LIN1 and LIN2

The following two rules are based on the LIN–ET procedure proposed in (Ow & Morton, 1989) for the weighted single machine problem. These rules, which will be denoted by LIN1 and LIN2, choose, at each iteration, the job with the largest value of the following priority indexes:

$$LIN1_j(S) = \begin{cases} \frac{1}{P_j(S)} & if \quad s_j(S) \leq 0 \\ \frac{1}{P_j(S)} - \frac{s_j(S)}{slk\_thr} \times \frac{2}{P_j(S)} & if \quad 0 < s_j(S) < slk\_thr \\ -\frac{1}{P_j(S)} & if \quad s_j(S) \geq slk\_thr \end{cases}$$

and

$$LIN2_j(S) = \begin{cases} \frac{1}{P_j(S)} & if \quad s_j(S) \leq 0 \\ \frac{1}{P_j(S)} - s_j(S) \times \left(\frac{1}{slk\_thr \times P_j(S)} + \frac{1}{P_j(S)}\right) & if \quad 0 < s_j(S) < slk\_thr \\ -\frac{s_j(S)}{P_j(S)} & if \quad s_j(S) \geq slk\_thr \end{cases},$$

where slk_thr, which stands for "slack threshold", is a parameter meant to represent a value such that slacks which are greater or equal to that value are considered large.

In the first branch of the priority index, identical in both heuristics, a job is late or on time if scheduled next. When one or more such jobs exist, LIN1 and LIN2 select the job using a shortest time rule, in line with various heuristics for (earliness and) tardiness problems (Kenneth R. Baker, 1974; Ow & Morton, 1989; Smith, 1956).

In the third branch, the job has a large slack, and is quite early. If all jobs are quite early, LIN1 chooses the job using a longest time rule, again in line with several heuristics for early/tardy problems (Ow & Morton, 1989; Valente, 2007; Valente & Alves, 2005). LIN2, on the other hand, selects the job with the minimum slack per required time. Finally, the middle branch performs a linear interpolation between the priority values corresponding to $s_j(S) = 0$ and $s_j(S) = slk\_thr$. Such an interpolation was first performed in the LIN–ET procedure (Ow & Morton, 1989).

The slk_thr parameter is calculated as follows. At each iteration, the slack threshold is set equal to $slk\_thr = w \times (C_{max}^{LB}(S) - t)$, where $C_{max}^{LB}(S)$ is a lower bound on the completion time of the last job on the final machine (makespan), given the current schedule S, and $0 \leq w \leq 1$ is a user-defined parameter. The lower bound is calculated using an adaptation of the procedure proposed in (Taillard, 1993). Indeed, the procedure of (Taillard, 1993) assumes that all machines are available at time zero. Since a lower bound is calculated at each iteration of LIN1 and LIN2, it was necessary to adapt the procedure to deal with non-zero machine availability times, which occur as jobs are scheduled.

The lower bound $C_{max}^{LB}(S)$ is calculated as follows. Let $Bef\_M_i(S) = min_{j \notin S}\left(t + \sum_{k=1}^{i-1} p_{kj}\right)$ be a lower bound on the time needed before reaching machine i, since it considers the availability time on the first machine, plus the minimum, over all unscheduled jobs, of the sum of the processing times on the machines that precede i. Also let $TPT\_M_i(S) = \sum_{j \notin S} p_{ij}$ be the total processing time required by all unscheduled jobs on machine i. Furthermore, let $Aft\_M_i(S) = min_{j \notin S}\left(\sum_{k=i+1}^{m} p_{kj}\right)$ be a lower bound on the time required after machine i, since it considers the minimum, over all unscheduled jobs, of the sum of the processing times on the machines that follow i.

For each machine i, a lower bound on the makespan is then calculated as $C_{max}^{LB}\_M_i(S) = max(Bef\_M_i(S), t_i) + TPT\_M_i(S) + Aft\_M_i(S)$. In the original bound of (Taillard, 1993), all machine availability times were zero. Given a partial schedule, and/or machine availability times different from zero, two adaptations were required. The first was including the availability time of the first machine in $Bef\_M_i(S)$. The second was using the maximum between the bound on the time needed before reaching machine i and the availability time of this machine: $max(Bef\_M_i(S), t_i)$. The lower bound on the makespan $C_{max}^{LB}(S)$ is equal to the maximum of all machine lower bounds: $C_{max}^{LB}(S) = max_i\left(C_{max}^{LB}\_M_i(S)\right)$.

IMPROVEMENT PROCEDURE

As described earlier, unforced idle time is considered only on the final machine as way to reduce the earliness of jobs. Consider two jobs *j* and *k* that are sequenced adjacent to each other with job *j* immediately before job *k*. If job *j* is early and there is idle time between jobs *j* and *k* on the final machine, then unforced idle time will be inserted before the start of processing of job *j* on the final machine. If enough unforced idle time is inserted the idle time between jobs *j* and *k* could be eliminated and processing between the two jobs on the final machine would be a continuous block. It could be that after unforced idle time is inserted there could be several jobs that belong to a block that has continuous processing on the final machine with idle time before the first job in the block. It is possible that switching the order of jobs within the block will not cause the block of processing on the final machine to be disrupted because of the unforced idle before the first job of the block (so processing on machines 1 through M – 1 are not constraining). Therefore, a localized approach that looks at blocks of jobs as if scheduling on a single machine may be beneficial. That is what the improvement procedure attempts.

The improvement procedure examines adjacent jobs within each block that was formed after the insertion of unforced idle to see if improvements can be found by swapping jobs. Some properties from single machine scheduling with earliness and tardiness are considered in this procedure. Let jobs *j* and *k* be adjacent jobs in a sequence with job *j* preceding job *k* with no idle time between their processing on machine *M*. Also, let S_{jM} be the start time of processing for job *j* on machine *M*. The following three properties are used to determine if switching the pair of jobs should be attempted.
Property 1) $p_{jM} < p_{kM}$ and $d_j > C_{kM}$
Property 2) $p_{jM} > p_{kM}$ and $S_{jM} + p_{km} \geq d_k$
Property 3) $d_j > C_{jM}$ and $C_{kM} > d_k$
If any of the three properties holds a swap is made; if a better solution results the swap is kept, otherwise it is reversed. The first two properties are properties that would result in a better solution on a single machine if the swap was performed but because of the other machines needs to be checked in this problem.

COMPUTATIONAL TEST

The test of the proposed algorithms consisted of randomly generated problems of various sizes in terms of the number of jobs and number of machines and under various conditions of due date range and tightness.

Data and Performance Measures

The dispatching heuristic procedures described in an earlier section along with the improvement procedure were tested on problems of various sizes in terms of the number of jobs and number of machines for nine sets of distributions of due date range and tightness. Each problem set consists of 10 problems. The problems within a problem set have the same number of jobs and machines, and the due dates for the jobs are generated using the same distribution. Eight levels of number of jobs (*n*) to be scheduled were tested: *n = 15, 20, 25, 30, 40, 50, 75* and *100*. Three levels of number of machines (*M*) were tested: *M = 5, 10* and *20*. The processing times of the jobs for each machine were generated using a uniform distribution over the integers 1 and 100. Due dates were randomly generated using a uniform distribution over the integers *MS (1 – r – R/2)* and *MS (1 – r + R/2)*, where *MS* is an estimated makespan found for the problem using the makespan lower bound proposed in Taillard (1993), and *R* and *r* are two parameters called due date range and tardiness factors. Three levels of due date range (*R*) were tested: *R = 0.2, 0.6* and *1.0* and three levels of due date tightness (*r*) were tested: *r = 0.0, 0.2* and *0.4*. These levels of *R* and *r* result in nine sets of due date parameters for each *n* and *M* combination.

The procedures were coded in Turbo Pascal and were tested on a Dell Inspiron 1525 GHz Lap Top computer. The measure of performance used to evaluate the dispatching procedures for the problems is percentage deviation (*% Dev*) of the total earliness and tardiness of the solution generated by each procedure from the lowest total earliness and tardiness generated by the procedures. $\% Dev = [(Z_h - Z_B)/ Z_h] * 100$, where Z_B = the lowest total earliness and tardiness of the solutions generated by the six procedures, and Z_h = the total earliness and tardiness of the solutions generated by the dispatching heuristic procedures (EDD, MDD, LIN1, LIN2, SLK, SLKP).

Results of the Tests

Among the dispatching heuristics, the MDD was the best across all the problem sizes. It had a % Dev that was less than 8% for all problem sizes. The LIN1 procedure was consistently the worst. This was surprising because this heuristic was found to be the best for the problem when unforced idle time was not allowed by (Valente and Schaller, 2017). In the (Valente and Schaller, 2017) study, the MDD procedure was the third best.

For n = 50 and M = 10 the influence of the due date tardiness and range parameters was checked. The results showed that MDD was generally the best and was relatively consistent across the parameters but as due dates became tighter and the range of due dates was narrow LIN2's performance improved.

When the improvement procedure was included the results showed that MDD_I performed the best over most of the problem sizes. The results also showed that MDD_I was robust with respect to the due date tightness and range parameters.

CONCLUSIONS

In this paper, six dispatching heuristics were proposed for minimizing total earliness and tardiness in permutation flow shops when using unforced inserted idle time to reduce job earliness is allowed. An improvement procedure that can be used with any of the six dispatching procedures was also proposed. These procedures were tested on problems of various sizes in terms of the number of jobs and machines, and nine sets of distributions that determine the tightness and range of due dates. All of the procedures are very efficient and were able to generate solutions quickly for all the problem sizes tested.

The results showed that the Modified Due Date (MDD) consistently generates solutions with a lower total earliness and tardiness than the other procedures tested. This was also true after the improvement procedure was added to the dispatching procedures. This result was interesting as the MDD procedure was not as good as some of the other procedures (LIN1 and LIN2) when inserting unforced idle time was not considered. In the tests that were conducted in this research the parameters used for the LIN1 and LIN2 procedures were the ones that were found to work well when inserting unforced idle time was not considered. An investigation needs to be done to see if adjusting the parameters for these procedures can improve their results. An area of additional research would be finding improvement approaches that can be applied to the solutions generated by the dispatching procedures.

REFERENCES

Baker, K.R. (1974). Introduction to sequencing and scheduling. New York: Wiley.

Baker, K.R. & Bertrand, J.W.M. (1982). A dynamic priority rule for scheduling against due-dates. Journal of Operations Management*,* 3(1), 37-42.

Baker, K.R. and Scudder, G.D. (1990) 'Sequencing with earliness and tardiness penalties: a review', *Operations Research*, Vol. 38, pp.22–36.

Chandra, C. Mehta, P. and D. Tirupati, Permutation flow shop scheduling with earliness and tardiness penalties, *International Journal of Production Research*, 2009; 47: 5591 – 5610.

Davis, J. S. and J. J. Kanet, "Single-machine scheduling with Early and Tardy Completion Costs," *Naval Research Logistics,* 40, 85 – 101, 1993.

Fry, T. D., Armstrong, R. D., and J. H. Blackstone, "Minimizing weighted absolute deviation in single machine scheduling," *IIE Transactions,* 9, 445 – 450, 1987.

Hoogeveen, H. (2005) Multicriteria scheduling, European Journal of Operational Research, Vol. 167, pp.592–623.

Jackson, J.R. (1955). Scheduling a production line to minimize maximum tardiness. In: Management Science Research Project. Los Angeles: University of California.

Kanet, J.J. and Sridharan, V. (2000) 'Scheduling with inserted idle time: problem taxonomy and literature review', Operations Research, Vol. 48, pp.99–110.

Kim, Y. D., and C. A. Yano, "Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates," *Naval Research Logistics*, 41, 913 – 933, 1994.

Madhushini, N., Rajendran, C. and Y. Deepa, Branch-and-bound algorithms for scheduling in permutation flowshops to minimize the sum of weighted flowtime/sum of weighted tardiness/sum of weighted flowtime and weighted tardiness/sum of weighted flowtime, weighted tardiness and weighted earliness of jobs, *Journal of the Operational Research Society*, 2009; 40: 991 – 1004.

Moslehi, G., Mirzaee, M., Vasei, M., and A. Azaron, Two-machine flow shop scheduling to minimize the sum of maximum earliness and tardiness, *International Journal of Production Economics*, 2009; 122: 763 – 773.

M'Hallah, R. An iterated local search variable neighborhood descent hybrid heuristic for the total earliness tardiness permutation flow shop. *International Journal of Production Research*, 52 (13); 3802 – 19.

Ow, P.S. & Morton, T.E. (1989). The Single-Machine Early Tardy Problem. Management Science*,* 35(2), 177-191.

Panwalkar, S.S. & Iskander, W. (1977). Survey of Scheduling Rules. Operations Research*,* 25(1), 45-61.

Rajendran, C., Formulations and heuristics for scheduling in a kanban flowshop to minimize the sum of weighted flowtime, weighted tardiness and weighted earliness of containers, *International Journal of Production Research*, 1999; 37: 1137 – 1158.

Schaller, J.E. and J.M. Valente, "An evaluation of heuristics for scheduling a non-delay permutation flow shop with family setups to minimize total earliness and tardiness," *Journal of the Operational Research Society*, vol. 64, no. 6, p. 805, 2013.

Schaller, J.E. and J.M. Valente, "A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness," *The International Journal of Production Research*, vol. 51, no. 3, p. 772, 2013.

Smith, W.E. (1956). Various optimizers for single-stage production. Naval Research Logistics Quarterly*,* 3, 59-66.

Taillard, E., Benchmarks for basic scheduling problems, *European Journal of Operational Research*, 1993, 64: 278-285.

Valente, J.M.S. (2007). Dispatching heuristics for the single machine early/tardy scheduling problem with job-independent penalties. Computers & Industrial Engineering*,* 52(4), 434-447.

Valente, J.M.S. & Alves, R.A.F.S. (2005). Improved heuristics for the early/tardy scheduling problem with no idle time. Computers & Operations Research*,* 32(3), 557-569.

Valente, J. M. S., and J. E. Schaller, "Efficient Heuristics for the Permutation Flowshop Scheduling Problem with Earliness and Tardiness Costs," under review, 2017.

Vepsalainen, A.P.J. & Morton, T.E. (1987). Priority Rules for Job Shops with Weighted Tardiness Costs. Management Science*,* 33(8), 1035-1047.

Zegordi, S. H., K. Itoh, and T. Enkawa, A knowledgeable simulated annealing scheme for the early/tardy flow shop scheduling problem, *International Journal of Production Research*, 1997; 33: 1449 – 1466.